CCS'S CONTROLLER-UNIQUE SOFTWARE

CALIFORNIA COMPUTER SYSTEMS
250 CARIBBEAN DRIVE
SUNNYVALE, CA 94086

89008-02600, rev C

First printing September 1980
Updated and revised January 1981
Current to revision four of the software

TABLE of CONTENTS

PREFACE

Your copy of CCS's controller-unique version of CP/M 2.2 is contained on the diskette shipped with this manual. CP/M is proprietary to Digital Research and can be used only under license. Please read the enclosed licensing agreement, fill out the registration card, and mail it to Digital Research. Being registered with Digital Research also ensures your receiving any enhancements or updates to CP/M directly from Digital Research. CCS's modifications to CP/M 2.2 are copyrighted; please fill out and return the enclosed registration card. Any enhancements or modifications of CP/M 2.2 for CCS product users will be sent to you. We suggest you make a backup of the system diskette as soon as possible; a label for your backup diskette, complete with serial number and the required copyright information, is included in the system diskette's envelope.

CHAPTER 1
INTRODUCTION

This manual describes California Computer System's modifications and additions to Digital Research's distribution version of CP/M 2.2. The standard distribution version of CP/M is designed to work with an Intel Microprocessor Development System. We have made the necessary modifications for the operating system to work with CCS's Model 2810 Z-80 CPU and Model 2422 Floppy Disk Controller set. In addition, we have expanded the capabilities of the standard version of CP/M 2.2 so that it supports 5¼" single-sided drives and 8" double-sided drives as well as 8" single-sided drives. It also supports both single- and double-density diskettes conforming to the IBM 3740 and System 34 formats and containing either 128, 256, 512, or 1024 bytes per sector.

The extensively modified or unique portions of the software include the system loader, the BIOS, and, to support the new capabilities of the BIOS, several of the utility programs. California Computer Systems' system loader, CCBOOT, is described in section 1.1 below, while CCS's customized BIOS and special utilities programs are described in Chapters 2 and 3. To aid you in further customizing the software, the Appendix includes application notes to provide you with the information you need to tailor the software to your system. In both the main chapters and the application notes we have assumed that you are familiar with the basic principles and terminology of CP/M. You may wish to read Digital Research's manual "An Introduction to CP/M Features and Facilities" if you are not familiar with CP/M.

1.1 CCBOOT

During a cold-start entry into CP/M, CCBOOT, contained in Sector 01, Track 00 of the system diskette, is loaded into system memory at locations 080h-0FFh by the ROM-resident bootstrap loader. The bootstrap loader actually reads in the first two sectors of Track 00, allowing those users who wish to expand CCBOOT to be able to do so without having to modify

the monitor ROM. During a cold-start, then, the contents of the second sector will write over the first 128 bytes of the TPA (0100h-017Fh). Once loaded, CCBOOT begins execution and loads into memory the system tracks (TRK 00 and 01 on an 8" diskette; TRK 00, 01, and 02 on a 5¼" diskette) of any IBM-compatible system diskette. It then transfers control to the cold-start routine in the BIOS. Please note that CCBOOT requires a 4 MHz system to read double-density 8" diskettes. (It can read double-density 5¼" diskettes in a 2 MHz system.) Neither CCBOOT nor the ROM-resident bootstrap loader is 8080-compatible; if you plan to use either in an 8080 system, the Z-80 unique instructions must be changed to their 8080 equivalents.

1.2 SYSTEM MEMORY CONFIGURATION

The system load in memory is organized as shown in your Digital Research Manual "CP/M 2.2 Interface Guide." However, the base of the system load, CBASE, is negatively offset 2K compared to the normal CP/M load to allow for the enlarged BIOS. Thus while CBASE in Digital Research's 20K distribution version of CP/M 2.2 is at 3400h, it is at 2C00h in CCS's 20K version. For each size memory configuration, then, the space available for the TPA in CCS's version is 2K less than in a normal CP/M load. This is true even if you are running a transient program with the CCP overlaid; the beginning address of the system load in this case, FBASE, is also negatively offset 2K from the standard CP/M load. Please note that the entry points into the BIOS will also be offset by 2K, or 800h.

1.3 RESERVED PAGE 0 RAM

Page 0 (00h-0FFh) in a CP/M system is reserved for system parameters. CCS's version of CP/M uses some unique parameters in addition to the standard CP/M parameters. Table 1-1 on the following page lists the page 0 locations used by CCS's version of CP/M 2.2.

Address	Contents
0000h-0002h	These locations contain a jump instruction to the warm-start entry point, 4203h+b.
0003h	The Intel Standard IOBYTE is stored in this location. It is set to its default value by CCBIOS during cold-starts and preserved during warm-starts.
0004h	The current default drive number is stored in this location.
0005h-0007h	These locations contain a jump instruction to the BDOS. The address contained in 0006-0007h is the beginning address of the BDOS and hence the lowest address in memory used by CP/M if the CCP is overlaid.
0008h-000Ah 0010h-0012h 0018h-001Ah 0020h-0022h 0028h-002Ah 0030h-0032h 0038h-003Ah	Called by the Z-80 restart commands, these locations contain jump vectors to the restart error routine in CCBIOS. They are loaded by CCBIOS during cold-starts and can be over-written by restart routines. During warm-starts, the contents are preserved. Locations 0038-003Ah contain a jump instruction into the the DDT when breakpoints are being used in the debug mode.
0040h-0053h	These locations contain disk parameters used by CCBOOT, STDBIOS, and CCBIOS. Locations 0040-004Fh are defined by CP/M as user scratch pad locations; 0050-0053h are held reserved, but are not used by present versions of CP/M. See Chapter 3 of your 2422 Owner's Manual for a description of these locations' contents.
005Ch-007Ch	The CCP uses these locations as a default file control block for the transient program.
007Dh-007Fh	These locations are the optional default random record position.
0080h-00FFh (0080h-017Fh)	Locations 080h-0FFh are a temporary buffer for CCBOOT when it is loaded into memory from the first sector of Track 00 of the system disk. Since the 2422's ROM-resident bootstrap loader reads in the first two sectors of Track 00, locations 0100h-017Fh are also written into.

Table 1-1 Page 0 Parameters

CHAPTER 2
CCBIOS

2.1 DESCRIPTION OF CCBIOS

CCBIOS is California Computer Systems' customized, expanded BIOS. Designed to work with the Model 2810 Z-80 CPU and Model 2422 Floppy Disk Controller, CCBIOS contains the console drivers and primitive disk routines necessary for a basic CP/M system. It implements the Intel MDS IOBYTE as described in Chapter 3 of the 2422 Owner's Manual and in Digital Research's "CP/M 2.2 Alteration Guide." In addition, CCBIOS supports both maxi and mini diskettes of varying formats by using a modified version of CP/M's DEBLOCK sector buffering algorithms. CCBIOS's increased capabilities, however, are bought at the expense of space; CCBIOS is too large to fit on the system tracks of a standard diskette formatted in 128-byte, single-density sectors. Since diskettes of this format are the accepted medium for data exchange, CCS decided to distribute CP/M on a standard diskette and to locate CCBIOS in the file RLOCBIOS.COM, which consists of CCBIOS, an executable relocator program, and a bitmap. The system tracks contain a limited BIOS, called STDBIOS. STDBIOS is a "standard" CP/M BIOS in that it works only with diskettes formatted in 128-byte single-density sectors. When CP/M is loaded into system memory from diskette, control is transferred to STDBIOS, which outputs the CP/M signon message and transfers control to the CCP. The CCP automatically loads RLOCBIOS.COM into the transient buffer area and transfers control to it. The relocator finds the location of the STDBIOS and lays CCBIOS over it, using the bitmap to add the correct bias to the jump vectors. CCBIOS then transfers control to the CCP, which outputs the CP/M prompt.

2.2 DISKETTE COMPATIBILITY

CCBIOS is designed to be compatible with 5¼" and 8" soft-sectored blank diskettes that conform to the IBM 3740 and System 34 standards for diskette format and that have 35 and 77 tracks per diskette, respectively. It supports double-sided 8" diskettes as well. Table 2-1 shows the compatible diskette formats.

DENSITY	BYTES PER SECTOR	DISKETTE SIZE	SECTORS PER TRACK	DISK CAPACITY[1]
Single	128	5.25"	18	73,728 bytes
		8"	26	249,600
Single	256	5.25"	10	81,920
		8"	15	288,000
Single	512	5.25"	5	81,920
		8"	8	307,200
Single	1024	5.25"	2	65,536
		8"	4	307,200
Double	128	5.25"	29	118,784
		8"	48	460,800
Double	256	5.25"	18	147,456
		8"	26	499,200
Double	512	5.25"	10	163,840
		8"	15	576,000
Double	1024	5.25"	5	163,840
		8"	8	614,400

[1] The disk capacity figures are based on single-sided diskettes, with 35 tracks in the case of mini diskettes, and reflect user capacity (the total disk capacity minus the system tracks).

TABLE 2-1 COMPATIBLE DISKETTE FORMATS

Track 00 of any diskette must be formatted in 128-byte, single-density sectors, as conforms to the IBM format standards. CCS's diskette initialization routine, CCSINIT, automatically formats Track 00 in 128-byte, single-density sectors and should be used to initialize your blank diskettes to ensure their compatibility. Note that unless you modify the software as described in Application Note 5, the system diskette must also be formatted in 128-byte, single-density sectors.

CCBIOS has been designed to be able to read recorded diskettes of the standard 128-byte sector, single-density, single-sided format. We cannot guarantee, however, that it can read diskettes of different formats recorded on other systems.

2.3 SYSTEM COMPATIBILITY

CCBIOS is designed to work in a system using the 2810 Z-80 CPU/2422 Disk Controller set. The basic I/O routines, I/O byte handlers, and primitive disk routines are essentially the same routines as described in Chapter 3 of your 2422 Owner's Manual. As with the 2422's MOSS 2.2 Disk Monitor, CCBIOS contains I/O driver routines only for the 2810's serial I/O port as console interface. Should you wish to add peripherals or change the console interface, see Application Note 1. If the AUTO BOOT jumper on the 2422 is set to ON, so that CP/M is automatically booted on reset, STDBIOS and CCBIOS initialize the 2810's serial port to 9600 baud. Note that CCBIOS, unlike CCBOOT, is 8080-compatible and can read and write double-density diskettes in a 2 MHz system.

The software is designed to control single-sided or double-sided 8" drives of the Shugart SA800/850 type or single-sided 5.25" drives of the Shugart SA400 type. For information on configuring your drives, see the 2422 Owner's Manual. Should you wish to use double-sided mini drives, a faster step rate, or mini drives containing more than 35 tracks, see the appropriate Application Notes.

2.4 SOFTWARE COMPATIBILITY

The deblocking capabilities of CCBIOS add considerable flexibility to your system. However, when the deblocking mode is used with certain CP/M programs, the last write to disk may be lost. These programs complete their action by writing to the disk and then "warm-booting" the system to return control to CP/M. Normally this causes no problem, since in the non-deblock mode data is transferred directly to the disk. But in the deblocking mode, data is transferred first to a memory buffer which holds the data until an event such as a subsequent read or write or directory update operation forces the contents of the buffer to be written out to disk. An occasional result of this holding operation is that a reboot operation can be initiated without the memory buffer being emptied first. Thus the final writes of data never get transferred from memory to disk and are lost. To ensure your software is compatible with the deblocking mode, you can do one of the following:

1. For new software, ensure that the final disk write operations are directory updates. CP/M is set up to immediately update the directory on the disk every time

128 bytes of directory information (one CP/M logical sector) are transferred. This effectively empties the buffer before a warm boot is initiated.

2. Or, after the final disk write in a routine, read any other physical sector. This will purge the buffer and ensure that the written data is transferred to the disk.

It is not a good idea to try to solve the problem by modifying the boot routine so that it clears the buffer before actually doing the warm start. Because a warm start is the usual technique for error recovery, you could aggravate the error condition further by attempting to write additional potentially bad data onto the diskette.

2.5 ERROR MESSAGES

CCBIOS contains four built-in error messages: Device Not Ready, Restart Error, I/O Assignment Error, and Cannot Boot. The Restart Error and I/O Assignment Error messages are essentially the same as the MOSS 2.2 Monitor error messages.

DEVICE NOT READY: If you try an operation with a drive which is not ready (i.e., the door is open) or nonexistent you will receive the error message

DRIVE N NOT READY

where N is the drive unit. To get out of the error condition, ready the drive (insert a diskette, shut the door) and hit any key on your console device. Your system will pick up from where it left off. If the problem is a nonexistent drive, you will have to reboot your system.

RESTART ERROR: A restart error occurs when a program does a restart to a location without restart handling code. The jump vectors loaded during cold-start initialization cause the CPU to jump to the restart error routine in CCBIOS which outputs the RST ERR message and returns control to CP/M.

I/O ASSIGNMENT ERROR: You can use the CP/M STAT command as described in the Digital Research manual "An Introduction to CP/M Features and Facilities" to alter the contents of IOBYTE and thus change the physical-to-logical I/O device assignments. CCBIOS contains driver routines only for the teletype

physical device; assigning any other physical device to a logical category forces a jump to the I/O Assignment Error routine when an I/O operation is performed involving that logical category. The I/O ASGT ERROR message is output, the I/O assignments default to the teletype for each logical category, and control returns to CP/M. For more information about the IOBYTE and the I/O Assignment Error, see Chapter 3 of your 2422 Owner's Manual. For information on patching peripheral drivers to CCBIOS, see Application Note 1.

BOOT ERROR: During a warm boot of CP/M, the warm-boot routine outputs the message

CANNOT BOOT

and rings the console bell if it cannot read the system after ten tries. If you hit the carriage return key, the routine will try ten more times to read in the system.

CHAPTER 3
CCS FLEXIBLE UTILITIES

3.1 CCSINIT

CCSINIT allows you to initialize unformatted diskettes or reformat already-formatted diskettes so that they conform to the IBM 3740 and System 34 standards and can be read by the 1793 disk controller chip. This routine formats diskettes in 128, 256, 512 and 1024 bytes per sector and inserts the needed ID field information for each sector. Once initialized, the diskettes can be read by the 179x family of disk controller chips and the 1771 disk controller chip. Appendix B of your 2422 Owner's Manual shows the format used by CCSINIT.

To be able to insert the proper ID field information, CCSINIT asks you questions about the disk to be initialized and prompts you for a reponse. If the response you enter is not a valid one, CCSINIT repeats the question. Entering a carriage return in reponse to a question kicks you out of CCSINIT and returns you to CP/M.

The following console dialog initializes a Shugart SA400-compatible 5.25" single-sided diskette in 128-byte, single-density sectors. (The user's entries are underlined.)

```
N>CCSINIT          [N = the drive unit]

CCSDISK FORMATTER PROGRAM 2.Ø
WHICH DRIVE (A-D)? B
SINGLE OR DOUBLE SIDED (Ø OR 1)? Ø
SINGLE OR DOUBLE DENSITY (S/D)? S
TRACK NUMBER (Ø-4C OR *)? *
SECTOR SIZE (Ø-3)? Ø
IS IT A MINI (Y/N)? Y
NUMBER OF TRACKS (23, 28, OR 46)? 23[space]
```

After the last question is answered, CCSINIT formats the diskette. If you specify a diskette format other than the standard 128-byte, single-density sectors, CCSINIT

automatically reformats the first track in 128-byte, single-density sectors to comply with the IBM format standard. Once the initialization procedure is complete, CCSINIT asks again the first question, WHICH DRIVE (A-D)?, to allow you to continue initializing diskettes. If you want to return to CP/M, simply enter a carriage return.

Should CCSINIT be unable to find track 00, it will return with a Seek Error message, specifying drive, diskette side, track, sector, and 1793 status. For example, the following error message shows a seek error on sector 00h, track 00h, side 0 of the diskette in drive B:

SEEK ERROR: DRIVE B; SIDE 00; TRACK 00; SECTOR 00; STATUS-E0

Should CCSINIT be unable to initialize a track, it returns an INIT ERROR message, containing the same information as the SEEK ERROR message. In the case of either type error, CCSINIT maintains control and returns to the beginning question.

The following questions need further elaboration:

SINGLE OR DOUBLE SIDED (0 OR 1)? -- If you choose double-sided (1), CCSINIT formats both sides of the diskette. This option supports a double-sided drive of the Shugart SA850 or SA450 type. The distribution version of CCBIOS does not support double-sided mini drives. Note that if you own revision A of the 2422 Floppy Disk Controller, selecting 1 may cause a diskette in a Per Sci drive to eject.

TRACK NUMBER (0-4C OR *)? -- One of the important facilities of CCSINIT is that it allows one track of a diskette to be formatted. Thus if you have a diskette with one bad track you can reformat the one track without losing the contents of the rest of the diskette. Or you can reformat TRK 00 in a non-standard format. (If you do so, CCBIOS, CCBOOT, and the MOSS 2.2 Monitor cannot read the diskette.) To format one track, enter the track number in hex followed by a space. Since 77 tracks (4Ch) are the maximum number of tracks on an 8" diskette, CCSINIT prompts for a number within the range of 0 to 4Ch. However, you can specify up to FFh. If the track number you specify is greater than the actual number of tracks on your diskette, you will receive a SEEK ERROR message when CCSINIT tries to format that track. The wildcard symbol, *, allows the entire diskette to be initialized.

SECTOR SIZE (0-3)? -- In this question, CCSINIT prompts for an answer in the form of the sector size code contained in the sector ID field: 0 equals 128 bytes, 1 equal 256 bytes, 2 equals 512 bytes, and 3 equals 1024. CCSINIT does not accept all four sector sizes for each density format; the IBM 3740

and System 34 formats do not allow 1024-byte, single-density sectors or 128-byte, double-density sectors. If you choose either of these sector formats, CCSINIT outputs the error message

BAD TRACK OR SECTOR SIZE/DENSITY SELECTION:

after the last question instead of formatting the diskette. It will return to the question SINGLE OR DOUBLE DENSITY (S/D)? to give you a chance to change your selections.

IS IT A MINI (Y/N)? -- CCSINIT assumes all 8" drives are set up for 77 tracks per diskette. However, mini drives differ in the number of tracks per diskette. If you answer N(o) to this question, CCSINIT skips the next question and proceeds to format the diskette. If you answer Y(es), CCSINIT asks you to specify the number of tracks on your mini diskette. It prompts for some of the common values: 23h for Shugart SA400 drives, 28h for MPI 51 and 52 drives, and 46h for MPI 91 and 92 drives. It will, however, accept any one or two digit hex value and try to format the number of tracks specified. If your response to the question TRACK NUMBER (0-4C OR *)? is greater than or equal to your reponse to the question NUMBER OF TRACKS (23, 28, OR 46)?, you will receive the error message

BAD TRACK OR SECTOR SIZE/DENSITY SELECTION:

and CCSINIT will start with the question SINGLE OR DOUBLE DENSITY (S/D)? again. Please note that you must accurately describe the diskette in your answer to IS IT A MINI (Y/N)?; the software uses the answer to the question to condition the 2422 for either 8" or 5.25" disk operations.

3.2 CCSYSGEN

The SYSGEN routine supplied with Digital Research's distribution version of CP/M works only with 8" diskettes formatted in 128-byte, single-density sectors. CCSYSGEN allows the system to be generated on 5.25" diskettes and supports the same diskette formats as CCBIOS. Note that if you wish to use a non-standard system diskette for CCS's version of CP/M, you must modify the software as described in Application Note 5. CCSYSGEN works directly with the 2422 Disk Controller board (it does not work through CCBIOS) and requires the CP/M load be configured for 24K or greater. Digital Research's standard SYSGEN is also available on the distribution diskette.

CCSYSGEN works essentially the same way as SYSGEN. The following console dialog generates the system residing on the diskette in drive A (source) onto the system tracks of the diskette in drive B (destination).

```
N>CCSYSGEN

CCS SYSTEM GENERATION PROGRAM 1.Ø
COPYRIGHT 198Ø CALIFORNIA COMPUTER SYSTEMS

SOURCE DRIVE: A
SOURCE ON A, THEN TYPE RETURN [cr]

DESTINATION DRIVE: B
DESTINATION ON B, THEN TYPE RETURN [cr]
```

If you specify the wrong source or destination drive, hit any key other than the carriage return key when CCSYSGEN asks for drive confirmation. You will be returned to CP/M. You may also respond to the request SOURCE DRIVE: with a carriage return if the system is already in memory. Should your version of CP/M be configured for a system smaller than 24K, you will receive the following message after specifying the source drive:

```
INSUFFICIENT MEMORY SPACE--NEED 24K
```

After you have entered a carriage return to confirm the destination drive, CCSYSGEN generates the system on the diskette in the destination drive. CCSYSGEN automatically determines drive size and diskette format. When it has finished formating the diskette, it gives you the warning message

```
WARNING: POSSIBLE LOST DATA
```

if the format of the source diskette gives it more capacity than the destination diskette and outputs the prompt

```
DESTINATION DRIVE:
```

to allow you to continue to generate system diskettes. Should you wish to return to CP/M, enter a carriage return. If CCSYSGEN is unable to complete the system generation, you will receive the error message

```
DISK I/O ERROR ON N
```

where N equals the drive unit A, B, C, or D.

3.3 MOVCPM

As mentioned in Chapter 1, CCS's distribution version of CP/M allocates 2K more of system memory for the BIOS than the standard Digital Research version. Thus CCS had to modify the Digital Research version of MOVCPM so that it would place the base of the system load 2K lower for each memory size. If you use the command

N>MOVCPM 64 *[cr] (cr = carriage return)

to reconfigure the system load for a memory capacity of 64K, MOVCPM locates CBASE at the same address used for CBASE by a standard Digital Research system configured for 62K.

You can find a description of the MOVCPM program in section 6.9 of the Digital Research manual "An Introduction to CP/M Features and Facilities." Please note, however, that CCS's version of MOVCPM does not support the first two commands listed in this section (commands MOVCPM cr and MOVCPM n cr) and will print out the message "SAVE 36 CPMxx.COM" instead of the message "SAVE 32 CPMxx.COM." But in all other respects it functions the same as Digital Research's MOVCPM.

3.4 GENMOD

Digital Research's GENMOD program is usually included with MP/M (Digital Research's multi-user operating system) and not with CP/M. However, since it is used to construct a bitmap for a relocatable file, such as the bitmap in RLOCBIOS.COM, CCS has obtained permission to distribute it with CP/M. Most users will not have any reason to use GENMOD; a few experienced programmers may wish to use it as described in Application Note 7.

To use GENMOD, create two hex files, one with an origin of Ø, the other with an origin of 100h. Concatenate the files using PIP. Enter

N>GENMOD file.hex file.com[cr]

where file.hex is the concatenated hex file and file.com is the name of the relocatable file you wish to create. GENMOD leaves in memory the COM file, containing the relative-Ø hex file followed by the bitmap.

3.5 CCCOPY

CCCOPY copies the contents of a diskette recorded by a disk controller chip of the member of the 179x family onto another diskette in a format compatible with CCBIOS. It reads a copies one track at a time, reading the track on the source diskette, formating the corresponding track on the destination diskette, and then writing the data to the formatted track. It examines closely the first three tracks (Tracks ØØ, Ø1, and Ø2) on the source diskette for data density and bytes per sector, duplicating their formats on the corresponding tracks of the destination diskettes. CCCOPY assumes that the rest of the tracks have the same format as Track Ø2.

If you specify the same diskette as the source and destination diskette, CCCOPY reformats the diskette, preserving the data. This facility can be useful for recovering data on diskettes with soft errors. However, to ensure against data loss, make a backup copy of the diskette before you attempt to reformat it with CCCOPY.

```
**************************************************************
*                                                            *
*     NOTE:  CCS assumes no liability for any data loss      *
*     resulting from the use of CCCOPY.                      *
*                                                            *
**************************************************************
```

CCCOPY cannot read diskettes recorded by a 1771 disk controller chip. To function, it requires systems containing 32K of memory and CCS's disk controller board since it makes direct calls to the board's registers. Both the source and destination drives must be the same size and the diskettes must be either both single-sided or double-sided. CCCOPY automatically determines what size drives are being used, and in the case of mini drives asks you to specify whether the diskettes are single or double-sided and the number of tracks they have per side.

The following example of a console dialog causes the contents of the single-sided mini diskette in drive B to be copied onto the single-sided mini diskette in drive C:

```
N>CCCOPY

CCS DISK COPY PROGRAM VERS 1.Ø

SOURCE DRIVE (OR CONTROL C TO ABORT) B
DESTINATION DRIVE (OR CONTROL C TO ABORT) C
IS IT DOUBLE-SIDED? N                                  [These lines
MINI DRIVES WILL DEFAULT TO 35 TRACKS.                   appear for
IF THIS IS O.K., TYPE A CARRIAGE RETURN.  OTHERWISE,       minis
ENTER THE NUMBER OF TRACKS: 4Ø[cr]                         only]
WARNING - CONTENTS OF DESTINATION DISKETTE WILL BE LOST.
 DO YOU WISH TO CONTINUE? Y
COPYING TRACK nn [track number in decimal]

SOURCE DRIVE (OR CONTROL C TO ABORT) [Control-C]
```

For each of the questions, CCCOPY accepts only reasonable responses; if it receives a clearly inappropriate answer, it returns to the first question in the dialog. The only exception is when it asks for the number of tracks (in decimal) on a mini diskette; if it receives any character other than a number or a carriage return, it will prompt for a correct response. A Control-C in reponse to either of the first two questions returns control to CP/M.

ERROR MESSAGES

BAD DRIVE COMBINATION--If you specify a a destination drive of a different size than the source drive, CCCOPY displays this error message and returns to the first question in the dialog.

BAD NUMBER - TRY AGAIN--This error message is displayed if you respond to the number of tracks query with any character other than a number or a carriage return. It gives you the opportunity to correct an erroneous response; should you accidently enter the wrong number, simply hit any key other than a number key or a carriage return. You will then get the opportunity to enter the number correctly.

DRIVE NOT READY -- If the source or destination drive is not ready (e.g., the drive door is open), this message is displayed and, if your terminal supports this feature, the terminal bell will ring. To allow CCCOPY to continue, ready the drive and then hit any key on the keyboard except Control-C. Control-C returns you to CP/M.

DRIVE IS WRITE PROTECTED - TYPE CR TO CONTINUE--This message is displayed if the destination drive is write protected. If you wish CCCOPY to continue, unprotect the diskette, return it to the drive, and hit the carriage return key. Any other response returns you to the first question in the dialog.

CANNOT READ SOURCE--If for some reason CCCOPY cannot read the source diskette, it displays this error message and returns to the first question in the dialog. Possible causes for this error message include an unformatted diskette or one formatted by a 1771 disk controller chip.

BAD SOURCE SECTOR nn - IGNORE?--If CCCOPY cannot read a sector on the source diskette after ten tries, it displays this error message, in which nn is the physical sector number, and asks whether or not you wish CCCOPY to continue. If you type Y, CCCOPY will transfer whatever was in the sector and continue copying the disk. Data integrity in this case can not be asssured. If the error was a soft error, the data is probably good. If it was a hard error, the resulting data is unpredictable. If you enter any character other than a Y, CCCOPY ceases its copy attempt and returns to the first query in the dialog. Since CCCOPY attempts to read a sector ten times before it responds with the error message, it is unlikely that it would be any more successful in subsequent attempts. If you discover that the same sector is in error on several contiguous tracks, it is likely that your diskette has been scratched or otherwise damaged.

WRITE ERROR - IGNORE? -- CCCOPY will try ten times to successfully format a track or write a sector on the destination diskette. After it fails the tenth time, it displays this message and asks if you wish it to continue. If you type Y, CCCOPY goes to the next sector or track and continues writing the rest of the diskette, thus effectively skipping over the bad area of the disk. If you enter any other character, CCCOPY returns to the first question in the dialog. Since CCCOPY has already attempted to write the sector ten times, it is unlikely it would be successful in any subsequent attempt.

APPLICATION NOTE 1

CUSTOMIZING CCBIOS

PURPOSE:

As mentioned in Chapter Two, CCBIOS contains drivers only for the 2810's serial port as console interface. If you wish to add peripherals, such as a line printer, or change the console interface, you will have to modify CCBIOS. This Application Note describes how to do so. It assumes that if you do not have the Model 2810 Z-80 CPU, you have a host computer on which to make the necessary modifications to STDBIOS and CCBIOS to drive your particular console interface. If you do not have a host computer, you will have to read the system tracks into your system, modify STDBIOS, and write the tracks back out onto disk using GETSYS and PUTSYS routines as described in the CP/M 2.2 Alteration Guide. The primitive disk routines in the MOSS 2.2 Disk Monitor can be used as a guide for writing GETSYS and PUTSYS or can be accessed themselves. After you have modified STDBIOS, CCBIOS can be customized as described below.

METHOD:

Whenever CCBIOS is entered through a call to a basic I/O routine (CONIN, CONOUT, CONST, LIST, PUNCH, READER, or LISTST), it jumps to one of the IOBYTE handling routines (CO, CI, CSTS, LO, PO, RI, LSTAT). The IOBYTE handler examines the IOBYTE to determine which of the four allowable physical devices is assigned to the logical category involved in the I/O operation. It then jumps to the driver routine for that particular physical device. Currently, only the teletype physical device routines TTYIN, TTYOUT, TTOST, and TTST contain driver code. At the end of the CCBIOS source listing the rest of the physical device routines are equated to the I/O Error routine, with the exception of TTYRDR and TTPNCH, which are equated to TTYIN and TTYOUT, respectively. To add a peripheral device, then, you must replace the appropriate equates with driver routines, keeping the routine label the same. For example, if you wish to add a line printer, eliminate equates

```
LPRT:   EQU   IOER
LPRTS:  EQU   IOER
```

and add routines LPRT and LPRST. Note that any list output routine require a corresponding list status routine and that every console input routine also requires a console status routine. See your CP/M 2.2 Alteration Guide for the CP/M calling conventions.

The teletype physical assignments are the default assignments for each logical category. Currently, the teletype device routines contain device-unique driver code for the 2810's serial port. Should you not have a 2810 CPU, you must modify the TTYIN, TTYOUT, TTOST, and TTST code in both CCBIOS and STDBIOS so that the routines drive your particular console interface. In addition, you need to change the BOOT routine code in both CCBIOS and STDBIOS to initialize the console interface.

The following steps allow you to modify CCBIOS and replace the original with your customized CCBIOS.

1. Load and modify CCBIOS.ASM using ED. In addition to adding or modifying the basic I/O routines, you may wish to modify the following equates at the beginning of the source listings in CCBIOS:

   a. MSIZE EQU 20 -- If you have a larger system memory than 20K, this line should be modified to reflect your system memory size in kilobytes.

   b. SBAUD EQU 12 -- If you wish to use the Auto Boot mode, you can specify a baud rate other than 9600 baud. Before the SBAUD line is a table giving the values of SBAUD for common baud rates. Note that if you do change SBAUD in CCBIOS you should change it in STDBIOS as well.

   c. MINI EQU TRUE; MAXI EQU TRUE -- If you have only one size drive, you can save space by eliminating code pertaining to the other size drive. To do so, change the TRUE statement to FALSE for the size drive you do not possess. STDBIOS can be similarly modified.

   You should also make any modifications regarding step rates, double-sided mini drives, etc. that you require. See the appropriate application notes. AFTER ASSEMBLY, YOUR MODIFIED CCBIOS SHOULD NOT EXCEED 2K BYTES IN MACHINE-EXECUTABLE CODE.

2. Assemble CCBIOS.ASM

3. Using DDT, load in RLOCBIOS.COM. The following console dialog loads in RLOCBIOS:

```
N>DDT RLOCBIOS.COM[cr]
DDT VERS 2.2
NEXT PC
ØBØØ Ø1ØØ
```

4. Next you must load CCBIOS.HEX in the BIOS portion of RLOCBIOS.COM. A memory map of RLOCBIOS is as follows:

Ø1ØØ-Ø1CFh	Relocator program (Do not modify!)
Ø1DØ-Ø9CFh	BIOS
Ø9DØ-ØACFh	Relocator bitmap

The DDT Read command loads hex files into memory at their assembled addresses, unless an offset is specified. To load CCBIOS.HEX, then, into location Ø1DØh, you must first calculate the necessary offset by subtracting the origin of CCBIOS.HEX from Ø1DØh. In the case of a 64K version of CP/M, for example, the offset would be ØFDØh. The following console dialog calculates the offset by using the DDT H command (undocumented in the DDT manual) which gives the sum and difference of any two hex addresses. It then loads CCBIOS.HEX into RLOCBIOS.

```
-HØ1DØ nnnn[cr]  (nnnn equals the origin of CCBIOS.HEX)
ssss dddd  (ssss and dddd equal the sum and difference)
-ICCBIOS.HEX[cr]
-Rdddd[cr]  (difference equals the offset)
```

5. Zero out the bitmap and save RLOCBIOS.COM. Since you have modified CCBIOS, the bitmap is no longer accurate, nor is it needed, since CCBIOS.HEX contains absolute addresses. The following console dialog performs these tasks:

```
-F9DØ,ACF,Ø[cr]   (zero out bitmap)
-GØ[cr]

N>SAVE 1Ø RLOCBIOS.COM[cr]
```

You now have a modified version of CCBIOS, configured for a specific memory size. If your system memory is larger than 2ØK, you now need to configure the operating system for your size memory. Assuming that you have no need to modify STDBIOS, you can use the MOVCPM command followed by CCSYSGEN to place a copy of CP/M, configured for your size memory, on the system tracks of an initialized diskette. Please note that your version of CP/M is fixed for a particular memory size and cannot be easily reconfigured. If you are an experienced programmer and wish to retain the relocating capability of RLOCBIOS, see Application Note 7.

If you have modified STDBIOS, you must patch your modified STDBIOS into the image of CP/M created by the MOVCPM command, as described in section 3 of your CP/M 2.2 Alteration Guide. (The image created by MOVCPM contains CCS's STDBIOS. Should you wish to replace the CCS STDBIOS image with your modified STDBIOS, allowing your customized version of CP/M to be easily reconfigured for different memory sizes, see Application Note 7.)

APPLICATION NOTE 2

DOUBLE-SIDED DRIVES

PURPOSE:

The 2422 Floppy Disk Controller supports double-sided mini and maxi drives. However, the current CCS distribution version of CP/M 2.2 does not support double-sided mini drives, since they do not provide a status signal to identify when a double-sided diskette is mounted. Thus any general software developed by CCS would have to implement either single- or double-sided mini diskette operations, but not both. Since single-sided drives are more common, CCS decided to implement single-sided operation.

There are two basic methods of implementing double-sided drive capabilities:

A. The reverse side of each diskette can be treated as an extension of the obverse side. In its easiest form, this method treats each track on side 1 as an extension of the corresponding track on side Ø; these associated tracks are often called cylinders. For example, a track on a 128-byte per sector, single-density 8" diskette would logically consist of sectors 1-26 on side Ø and sectors 27-52 on side 1;

B. Each side of the diskette can be treated as a separate logical unit of assignment; that is, each physical drive is treated as two logical drives.

The current version of CCBIOS uses method A to support 8" double-sided drives. Those who wish only to implement double-sided mini drives will also want to use method A, since CCBIOS already carries the code for it. However, there may be some users who wish to implement method B for double-sided 8" and 5.25" drives.

METHOD A

To implement method A for double-sided mini drives, only the 5.25' disk parameter tables (DP5xx) need to be changed to accomodate the increased capacity of double sided drives. The new values for these tables can be defined by using the CP/M DISKDEF macro to suit your individual desires or you can use the following suggested values:

	TABLES DP5xx					
Parameters	SØ	S1	S3	DØ	D1	D2
Sectors Per Track	36	4Ø	32	58	72	8Ø
Block Shift Factor	3	3	3	3	4	4
Block Mask	7	7	7	7	15	15
Extent Mask	Ø	Ø	Ø	Ø	1	1
Blocks Per Disk	143	159	127	231	143	159
No. Dir Entries	63	63	63	63	63	63
ALLOC Ø	192	192	192	192	128	128
ALLOC 1	Ø	Ø	Ø	Ø	Ø	Ø
Dir Check Vector Size	16	16	16	16	16	16
System Track Offset	3	3	3	3	3	3

Note: For disks formatted in 512-byte, single-density sectors and in 1Ø24-byte, double-density sectors, the software uses Tables DP5S1 and DP5D2, respectively.

Table A-1 Double-sided Mini Parameters

We arrived at these values by using the DISKDEF Macro and judgement. For instance, we chose the block size (CP/M's logical allocation data size increment) so that every block on the diskette could be defined by 8-bit value. Similarly, we selected a number for directory entries sufficient for the diskette capacity.

METHOD B

The following implementation applies to CCBIOS. STDBIOS can be similarly modified if you desire.

Method B requires that the routine SELDSK be modified to perform the following translation:

Value of Register C on Entry	Drive Select	Side Select
00	00	00
01	00	01
02	01	00
03	01	01
04	02	00
05	02	01
06	03	00
07	03	01

Table A-2 SELDSK Translations

Below is a modified version of routine SELDSK. This modification has a ripple effect: Routine WBOOT and Tables PRMTBL and DPBASE must be modified to allow for 8, instead of 4, disks. DPBASE in turn requires more scratch RAM to be dedicated to it. These modifications are also shown below. This implementation of Method B adds 93 bytes of initialized RAM and 28Ø bytes of uninitialized RAM to the BIOS requirements.

Routine SELDSK Modifications:

```
SELDSK:     LXI     H,Ø
            MOV     A,C
            CPI     8
            RNC
SELDSK1:    MOV     L,A
            DAD     H
            DAD     H
            DAD     H
            DAD     H
            XCHG
            LXI     H,DPBASE
            DAD     D
CKSET:      PUSH    H
            MOV     A,C
            ADD     A
            LXI     H,PRMTBL+1
            MOV     E,A
            DAD     D
            MOV     A,M
            ORA     A
            STA     SEKSEL
            JNZ     SET4A
            XCHG
            INR     L
            SHLD    LUNIT
```

```
                LHLD    DISKNO
                XTHL
                PUSH    H
                LHLD    SECTOR
                XTHL
                PUSH    H
                MOV     A,C
                RAR
                STA     DISKNO
                STA     SEKDSK
                MVI     A,ØDØH
                JNC     CKSET1
                MVI     A,Ø9ØH
CKSET1:         STA     SIDE
                PUSH    B
                PUSH    D
                CALL    IDRD
                POP     H
                DCX     H
                LDA     STPRAT
                MOV     M,A
                INX     H
                LDA     CUNIT
                STA     SEKSEL
                MOV     M,A
                LXI     D,4
                LXI     H,MSELTBL
                MOV     B,A
                ANI     1ØH
                JZ      SETUP1
                LXI     H,SELTBL
SETUP1:         MOV     A,B
                ANI     4ØH
                JZ      SETUP2
                DAD     D
                DAD     D
SETUP2:         LDA     IDSV+3
                CPI     2
                JC      SET3
                DAD     D
SET3:           XCHG
                POP     H
                XTHL
                PUSH    H
                LDAX    D
                MOV     M,A
                INX     D
                INX     H
                LDAX    D
                MOV     M,A
                INX     D
```

```
                PUSH    D
                LXI     D,9
                DAD     D
                POP     D
                LDAX    D
                MOV     M,A
                INX     D
                INX     H
                LDAX    D
                MOV     M,A
                POP     H
                POP     B
                XTHL
                SHLD    SECTOR
                POP     H
                XTHL
                SHLD    DISKNO
SET4:           POP     H
                MOV     A,C
                RAR
                STA     SEKDSK
                MVI     A,ØDØH
                JNC     SET5
                MVI     A,Ø9ØH
SET5:           STA     SEKSID
                MOV     A,C
                RET
```

Warm Boot Modifications:

```
WBOOT:          LXI     SP,TBUF
                .
                .
                MVI     B,16        ;ZERO OUT PRMTBL
                LXI     H,PRMTBL
LOADlA:         MOV     M,A
                .
```

Table DPBASE Modifications:

```
DPBASE          DW      Ø,Ø
                .
                .
;DISK PARAMETER HEADER FOR DISK Ø4
                DW      Ø,Ø
                DW      Ø,Ø
                DW      DIRBF,Ø
                DW      CHK04,ALL04
```

```
;DISK PARAMETER HEADER FOR DISK Ø5
                DW      Ø,Ø
                DW      Ø,Ø
                DW      DIRBF,Ø
                DW      CHKØ5,ALLØ5
;DISK PARAMETER HEADER FOR DISK Ø6
                DW      Ø,Ø
                DW      Ø,Ø
                DW      DIRBF,Ø
                DW      CHKØ6,ALLØ6
;DISK PARAMETER HEADER FOR DISK Ø7
                DW      Ø,Ø
                DW      Ø,Ø
                DW      DIRBF,Ø
                DW      CHKØ7,ALLØ7
```

Table PRMTBL Modifications:

```
PRMTBL:         DB      Ø,Ø         ;DRIVE Ø STEP RATE,SELECT BYTE
                DB      Ø,Ø         ;       1
                DB      Ø,Ø         ;       2
                DB      Ø,Ø         ;       3
                DB      Ø,Ø         ;       4
                DB      Ø,Ø         ;       5
                DB      Ø,Ø         ;       6
                DB      Ø,Ø         ;       7
```

Additional Scratch Ram for DPBASE:

```
ALLØ4:          DS      38
ALLØ5:          DS      38
ALLØ6:          DS      38
ALLØ7:          DS      38
CHKØ4:          DS      32
CHKØ5:          DS      32
CHKØ6:          DS      32
CHKØ7:          DS      32
```

APPLICATION NOTE 3

STEP RATES

PURPOSE:

The step rate of a drive's read/write head differs from drive to drive. For example, Shugart SA8ØØ drives have an 8 ms step rate, Shugart 85Ø drives have a 3 ms step rate, and Memorex 55Ø drives have a 6 ms step rates. CCBIOS implements step rates of 6 ms for 8" single-sided drives and 3Ø ms for both sizes 5.25" drives. Although these rates work with almost all drives, they may make for noisy operation if your drive can support faster rates. This Application Note identifies the necessary modifications to CCBIOS to alter the step rates. Note that CCBIOS also implements 3 ms step rates for 8" double-sided drives, but that rate should work with all double-sided drives. There is no easy way of altering the 8" double-sided step rate.

METHOD:

The step pulses to the drive are issued by the 1793 disk controller chip at the rate specified by the last two bits in any head movement command. These bits must be altered to specify a different step rate. To make alteration easier, CCBIOS builds up each head movement command by adding the step rate bits to the basic command. The hex values of these step bits correspond to the following step rates:

STEP RATE BITS	STEP RATE	
	8"	5¼"
ØØh	3ms	6ms
Ø1h	6ms	12ms
Ø2h	10ms	20ms
Ø3h	15ms	30ms

Table A-3 Step Rate Values

If you are modifying the CCBIOS source code, you can change the step rates by changing the value of the STEP5 and STEP8 equates at the beginning of the source code.

If you wish to change the step rates in the machine executable code, load in file RLOCBIOS.COM, using DDT. The actual locations you will have to change depend on the serial number of your distribution diskette. The following console dialog will change the step rates in the current version of CCBIOS:

```
N>DDT RLOCBIOS.COM[CR]
DDT VERS 2.2
NEXT PC
ØBØØ Ø1ØØ

-S65F[CR]
Ø65F Ø3 nn[CR]  (nn = desired mini step rate bits)
Ø66Ø D2 .[cr]
-S671[CR]
Ø671 Ø1 nn[CR] (nn = desired maxi step rate bits)
Ø672 2B .[cr]
-GØ

N>SAVE 1Ø RLOCBIOS.COM  (Do not rename the file!)
```

Since STDBIOS is transitory, no change to its step rates needs to be done.

APPLICATION NOTE 4

EXTENDED-TRACK MINI DRIVES

PURPOSE:

The distribution version of CCBIOS supports 8" drives with 77 tracks and 5.25" diskettes with 35 tracks. However, some mini drives have more than 35 tracks available: the MPI B-51/52, Tandon TM 100, and QUME Data Track 5 have 40 tracks, while the MPI B-91/92 have 80 tracks. This application note identifies the necessary CCBIOS changes to take advantage of the these drives' added capacity.

METHOD:

In the case of single-sided drives, only the disk parameter tables need to be changed in CCBIOS and, if you wish, STDBIOS. The necessary changes to the tables are shown below. In the case of the double-sided drives, the changes identified in Method A of Application Note 2 for installing double-sided drives also need to be made. In addition, some of the drives can support a faster step rate; see Application Note 3.

	Drives:	SINGLE-SIDED		DOUBLE-SIDED	
	Tracks:	40	80	40	80
DP5S0:	DW	18	18	36	36
	DB	3	3	3	4
	DB	7	7	7	15
	DB	0	0	0	1
	DW	82	172	165	172
	DW	63	63	63	63
	DB	192	192	192	128
	DB	0	0	0	0
	DW	16	16	16	16
	DW	3	3	3	3

```
DP5S1:  DW    20    20    40    40
        DB     3     3     3     4
        DB     7     7     7    15
        DB     0     0     0     1
        DW    91   192   184   192
        DW    63    63    63    63
        DB   192   192   192   128
        DB     0     0     0     0
        DW    16    16    16    16
        DW     3     3     3     3
DP5S3:  DW    16    16    32    32
        DB     3     3     3     4
        DB     7     7     7    15
        DB     0     0     0     1
        DW    73   153   147   153
        DW    63    63    63    63
        DB   192   192   192   128
        DB     0     0     0     0
        DW    16    16    16    16
        DW     3     3     3     3
DP5D0:  DW    29    29    58    58
        DB     3     4     4     5
        DB     7    15    15    31
        DB     0     1     1     3
        DW   133   138   133   138
        DW    63    63    63   127
        DB   192   128   128   128
        DB     0     0     0     0
        DW    16    16    16    32
        DW     3     3     3     3
DP5D1:  DW    36    36    72    72
        DB     3     4     4     5
        DB     7    15    15    31
        DB     0     1     1     3
        DW   165   172   165   172
        DW    63    63    63   127
        DB   192   128   128   128
        DB     0     0     0     0
        DW    16    16    16    32
        DW     3     3     3     3
DP5D2:  DW    40    40    80    80
        DB     3     4     4     5
        DB     7    15    15    31
        DB     0     1     1     3
        DW   184   192   184   192
        DW    63    63    63   127
        DB   192   128   128   128
        DB     0     0     0     0
        DW    16    16    16    32
        DW     3     3     3     3
```

Note: The disk parameter tables DP5S1 and DP5D2 are also used for 512-byte, single-density and 1024-byte double-density diskettes, respectively.

APPLICATION NOTE 5

NON-STANDARD SYSTEM DISKETTES

PURPOSE:

As mentioned in Chapter 2, the standard BIOS, STDBIOS, located on the system tracks can handle diskettes formatted in only 128-byte, single-density sectors. Since the loading of RLOCBIOS requires the disk routines in STDBIOS, the distribution version of STDBIOS is useless on the system tracks of a non-standard diskette. The solution is to put CCBIOS, which contains the code necessary to read non-standard diskettes, on the system tracks, replacing STDBIOS.

METHOD:

1. You must first load in from the system diskette and modify CCBIOS.ASM, using ED, for any system unique features you wish to install, such as peripheral drivers, etc. See Application Note 1 for information on how to modify CCBIOS to support the features you want. Be sure the MSIZE line in the source code reflects your memory size in kilobytes (decimal). Please note that your modified BIOS should not exceed E00h (3584) bytes, including uninitialized work space.

2. Assemble the file.

3. Scan the assembled output listing for the start address and the last address for code or data (uninitialized BIOS or workspace RAM is not included). Subtract the two addresses and compare the result to the value in the following table showing the space available to the BIOS for your chosen diskette format:

DENSITY	BYTES PER SECTOR	MAXIMUM BIOS LENGTH (BYTES)[1]	
		8" DISK	5.25" DISK
Single	128	380h (896)	480h (1152)
	256	580h (1408)	680h (1664)
	512	680h (1664)	680h (1664)
Double	256	1080h (4224)	1680h (5760)
	512	1480h (5248)	1A80h (6784)
	1024	1680h (5760)	1A80h (6784)

[1] The above values assume that two tracks (TRK 00 and 01) are dedicated to system use on 8" diskettes and that three tracks (TRK 00-02) are dedicated to system use on 5.25" diskettes. They also assume that the system loader requires only one sector. CCBOOT (already installed on the system tracks) meets this requirement and will load any of the diskette formats above.

Table A-4 Allowable BIOS Lengths

If your customized BIOS is greater than the BIOS space available on the system tracks, you will have to reduce its size. Since CCBIOS was devised to work with a wide variety of systems and applications, it is undoubtedly carrying code you do not need. For example, if yc are not using mini drives, you can eliminate code used by mini drives only. Once your customized BIOS is equal to or less than the BIOS space available, then you can move CCBIOS onto the system tracks so that it replaces STDBIOS. The remainder of this application note explains how to do so.

4. Initialize the new system diskette to the selected format by using CCSINIT.

5. Create an interim system load on the new system diskette (note that this system cannot be booted!)

   a. Run MOVCPM n * where n equals the system size you are creating.

   b. Save the resultant system on the diskette:

      N>SAVE 36 CPMn.COM[cr]

6. Install the new CCBIOS by using DDT. Unless an offset is specified, the R command will load CCBIOS.HEX at its origin; thus you must calculate the offset necessary to load CCBIOS.HEX into position at 1F80h by subtracting the origin of CCBIOS.HEX from 1F80h. The following console dialog calculates the offset by using the DDT H command (undocumented in the DDT manual) which gives the sum and

difference of any two hex numbers. It then loads CCBIOS.HEX into position.

```
N>DDT CPMn.COM[cr]
DDT VERS 2.2
NEXT PC
2900 0100

-H1F80 nnnn[cr]   (nnnn equals CCBIOS origin)
ssss dddd  (ssss and dddd are the sum and difference)
-ICCBIOS.HEX[cr]
-Rdddd[cr]  (difference equals offset)
```

7. While still in DDT, suppress the RLOCBIOS autoload feature:

```
-S987[cr]
0987 08 00[cr]
0988 52 .[cr]
```

8. Also, while in DDT, you should change the BIOS end address parameter in CCBOOT if your BIOS is over 600h bytes. CCBOOT resides at locations 900 through 97F; the location of the BIOS end address parameter is 929h. The parameter is the first two hex digits of the BIOS end address. It can be changed to reflect a BIOS length of up to E00h in 256 byte increments. For example, if you have a 64K version of CP/M with a BIOS of 90Ah bytes, the BIOS end address should be FB0Ah. Rounded up to the nearest 256-byte boundary, the end address becomes FC00h. Change the end address parameter, then, from F8 to FC:

```
-S929[cr]
0929 F8 FC[cr]
092A 78  .[cr]
```

If you plan to modify the source code for CCBOOT, be sure to change the 600H in the line

```
LXI   D,(BIOS+600H)+2
```

to reflect your BIOS length in 256-byte blocks. Remember to round up.

9. Get out of DDT with a G0[cr] command and then save your copy of CP/M for reference or further alterations.

```
N>SAVE 41 CPMnD.COM
```

10. Write the completed system load onto the new system disk using CCSYSGEN, skipping the source drive request.

11. Cold boot the disk and you are on your way.

APPLICATION NOTE 6

SECTOR SKEW

PURPOSE:

Some users may wish to modify the sector skew values to increase disk access speed or to gain compatibility with diskettes recorded on different systems. To do so, you must modify the sector skew tables in CCBIOS. In the current version of CCBIOS, the sector skew tables are generated by using an algorithm which may not be immediately obvious to a user wishing to modify the tables. This Application Note describes how to modify the tables.

METHOD:

To change the sector skew values, you need to change two values in the skew table for a diskette of a particular size and format. The current sector skew tables are as follows:

Table A-5

Sector Skew Values

```
T848    DB    48,48,1
T826    DB    26,13,6
T815    DB    15,15,4
T88     DB    8,8,3
T84     DB    4,2,2

T529    DB    29,29,7
T518    DB    18,9,4
T51Ø    DB    1Ø,1Ø,3
T55     DB    5,5,2
T52     DB    2,2,1
```

The first value in each sector skew table is the number of sectors per track (one side only) and remains constant. The third value is the sector skew factor and can be modified to any number between 1 and the first value. The second number is the result of the first value divided by the greatest common denominator (GCD) of the first and third values. It must be changed if the new skew factor results in a different GCD. For example, if you wanted to change the skew factor in table T51Ø to 5, the second value must change to 2 (GCD=5; 1Ø/5=2). Of course, CCBIOS must be reassembled and installed (see Application Note 1 for procedure).

APPLICATION NOTE 7

BUILDING A RELOCATABLE BIOS

PURPOSE

FOR EXPERIENCED PROGRAMMERS ONLY.

RLOCBIOS.COM contains, as described in section 2.1, a relocator program, CCBIOS, and a bitmap used by the relocator program. If you customize CCBIOS as described in Application Note 1, the bitmap will no longer be accurate and thus RLOCBIOS loses its ability to relocate CCBIOS at any even page. In addition, MOVCPM.COM contains a relocator program, an image of CP/M (containing STDBIOS), and a bitmap. Unless MOVCPM is modified, it will contain the original CCS STDBIOS. Thus if you have modified STDBIOS, you may wish to patch your modified STDBIOS into MOVCPM and rebuild the bitmap so that you can easily reconfigure CP/M for different memory sizes.

This Application Note identifies how to rebuild the bit map for STDBIOS or CCBIOS and to patch the modified BIOS and bitmap into their respective relocating program. GENMOD is used to build the bitmap; please review its description in section 3.4.

METHOD:

STDBIOS BUILD

1. Create two source files of STDBIOS: STDBIOSØ.ASM and STDBIOS1.ASM. Set the origin of the CCP in STDBIOSØ.ASM equal to ØØØØh; set it in STDBIOS1.ASM to Ø1ØØh.

2. Assemble the files.

3. Concatenate the hex files, using PIP:

```
N>PIP STDBIOS.HEX=STDBIOSØ.HEX,STDBIOS1.HEX
```

4. Now use GENMOD to create a new file, STDBIOS.COM, which contains STDBIOS and the bitmap.

```
N>GENMOD STDBIOS.HEX STDBIOS.COM
```

To be consistent with the MOVCPM program which loads the image of CP/M in the TPA at 900h, GENMOD uses address 900h as the origin for the file it leaves in memory. Since the beginning of the BIOS is offset 1600h from the beginning of the CCP, STDBIOS.COM now resides in memory at 1F00h.

5. Save STDBIOS.COM on disk:

```
N>SAVE 37 STDBIOS.COM
```

6. Using the DDT, load in MOVCPM.COM in preparation for moving the new BIOS and bitmap into place:

```
N>DDT MOVCPM.COM
```

MOVCPM.COM now resides at 100h, with CCS's original STDBIOS beginning at 2000h and its bitmap at 27E0h.

7. Read in the new STDBIOS.COM, specifying a bias of 4000h to load it into uncontested memory:

```
-ISTDBIOS.COM
-R4000
```

STDBIOS.COM now resides at 5F00h (1F00h + 4000h).

8. Find the beginning of the bitmap; it will vary according to the length of your BIOS. Each bit of the bitmap represents one byte of the BIOS; a bit is set to 1 only if its corresponding byte differed in the relative-0 and relative-1 versions of the hex file. Since STDBIOS begins with a series of jump targets, the beginning of the bitmap bitmap will show a repeating bit pattern of two zeros followed by a 1. (Since the two hex files were offset by 100h, only the high order address byte of a jump vector differ between the files.) In a hex dump, this pattern will appear as the following three byte sequence, repeating at least twice: 24h-92h-49h. Once you have found the beginning of the bitmap, move STDBIOS and the bitmap into position at 2000h and 27E0, respectively. The following example assumes that bitmap is located at 6518h and ends at 6587h:

```
-M5F00 627F 2000  (move STDBIOS into position)
-M6518 6587 27E0  (move bitmap into position)
-G0
```

9. Save your new MOVCPM.COM:

```
N>SAVE 40 MOVCPM.COM
```

CCBIOS BUILD:

1. Read the instructions for building STDBIOS for an understanding of the general procedure.

2. Create CCBIOSØ.ASM and CCBIOS1.ASM as described in step 1 of STDBIOS build.

3. Assemble them; then concatentate them using PIP:

```
N>PIP CCBIOS.HEX=CCBIOSØ.HEX,CCBIOS1.ASM
```

4. Create the file CCBIOS.COM, containing CCBIOS and the bitmap, by using GENMOD and then save it on disk:

```
N>GENMOD CCBIOS.HEX CCBIOS.COM

N>SAVE 42 CCBIOS.COM
```

5. Get into the DDT and load CCBIOS.COM:

```
N>CCBIOS.COM
```

   CCBIOS.COM now resides at 1FØØh.

6. Read in RLOCBIOS.COM. No bias is necessary since it will load safely under CCBIOS.COM.

```
-IRLOCBIOS.COM
-R
```

   A memory map of RLOCBIOS.COM is as follows:

Ø1ØØ-Ø1CFh	Relocator program
Ø1DØ-Ø9CFh	Unmodified CCBIOS
Ø9DØ-ØACFh	Relocator bitmap

7. Move CCBIOS and bitmap into position. To do so, you will have to find the beginning of the bitmap using the procedure described in step 8 above. In the following example, the bitmap is located at 29CAh:

```
-M1FØØ 26FF Ø1DØ    (move CCBIOS into position)
-M29CA 2AC9 Ø9DØ    (move bitmap into position)
-GØ
```

8. Save your new RLOCBIOS.COM:

```
N>SAVE 1Ø RLOCBIOS.COM
```